EUMETSAT Satellite Application Facility on Climate Monitoring

CLAAS-2 Auxiliary Data User Guide

DOI: 10.5676/EUM_SAF_CM/CLAAS/V002

Reference Number	SAF/CM/DWD/AUX/SEV/CLD
Issue/Revision Index:	1.0
Date:	05.01.2017

Document Signature Table

	Name	Function	Signature	Date
Authors	Stephan Finkensieper	CM SAF scientist (DWD)		05/01/2017
Editor				
Approval				
Release				

Distribution List

Internal Distribution	
Name	**No. Copies**
DWD Archive	1
CM SAF Team	1

External Distribution		
Company	**Name**	**No. Copies**
PUBLIC		1

Document Change Record

Issue/Revision	Date	DCN No.	Changed Pages/Paragraphs
1.0	05/01/2017		

Table of Contents

List of Figures

Reference Documents

Reference	Title	Code
RD 1	Product User Manual SEVIRI Cloud Products Edition 2 (CLAAS-2)	SAF/CM/DWD/PUM/SEV/CLD/2.1

1 Introduction

The second edition of CM SAF's CLoud property dAtAset using Seviri (CLAAS-2) provides cloud properties derived from the SEVIRI imager onboard geostationary METEOSAT Second Generation (MSG) satellites. Two categories of products are available:

- **Level 2**: 15 minutes repeat cycle, native SEVIRI grid (3km NADIR resolution)
- **Level 3**: Daily and monthly composites on regular 0.05° and 0.25° grids

The spatial resolution of the different level 3 products is given in Table 1.1. For detailed information about the CLAAS-2 product suite, see the Product User Manual RD 1.

Product	**Resolution**
Daily Mean	0.05°
Monthly Mean	0.05°
Monthly Mean Diurnal Cycle	0.25°
Monthly Histogram (CTO, LWP, IWP)	0.05°
Monthly Histogram (JCH)	0.25°

Table 1.1 Spatial resolution of CLAAS-2 level 3 products

In order to facilitate work with the CLAAS-2 dataset, we provide the following auxiliary data:

Parameter	**netCDF Variable Name**		
	Level 2	**Level 3, 0.05°**	**Level 3, 0.25°**
Latitude	`lat(y, x)`	`lat(lat)`	`lat(lat)`
Longitude	`lon(y, x)`	`lon(lon)`	`lon(lon)`
Land-Sea-Mask	`lsm(y, x)`	`lsm(lat, lon)`	/
Land Fraction	/	/	`land_fraction(lat, lon)`
Altitude	`alt(y, x)`	`alt(lat, lon)`	`alt(lat, lon)`
Satellite-Zenith-Angle	`satzen(lon0, y, x)`	`satzen(lon0, lat, lon)`	`satzen(lon0, lat, lon)`
Surface area covered by each pixel	`pixel_area(y, x)`	`pixel_area(lat, lon)`	`pixel_area(lat, lon)`
Dimensions	x/y=3636, lon0=4	lat/lon=3600, lon0=4	lat/lon=720, lon0=4

Table 1.2 Available auxiliary data and corresponding netCDF variable names. In the level 2 file, y and x denote the SEVIRI image rows and columns, respectively.

As the satellite position changes over time, Satellite-Zenith-Angles are available for the most prominent satellite positions (see section 4 for details).

2 Level 2 Auxiliary Data

CLAAS-2 level 2 products are distributed on the native SEVIRI grid, which is an irregular grid. Hence, latitude and longitude coordinates providing the georeferencing of each pixel are two-dimensional. Although the satellite position changes over time, the SEVIRI images are reprojected so that (lat,lon)=(0,0) is always in the image centre. That means the coordinates are valid throughout the dataset. However, this is not the case for satellite zenith angles, see section 4.

Land-Sea-Mask and altitude were taken from the SAFNWC-MSG cloud retrieval which was used in CLAAS-2. Pixel area and satellite zenith angle were directly derived from the latitude/longitude coordinates.

Auxiliary data for level 2 products are stored in the netCDF4 file `claas2_level2_aux_data.nc` . The following figures show its contents.

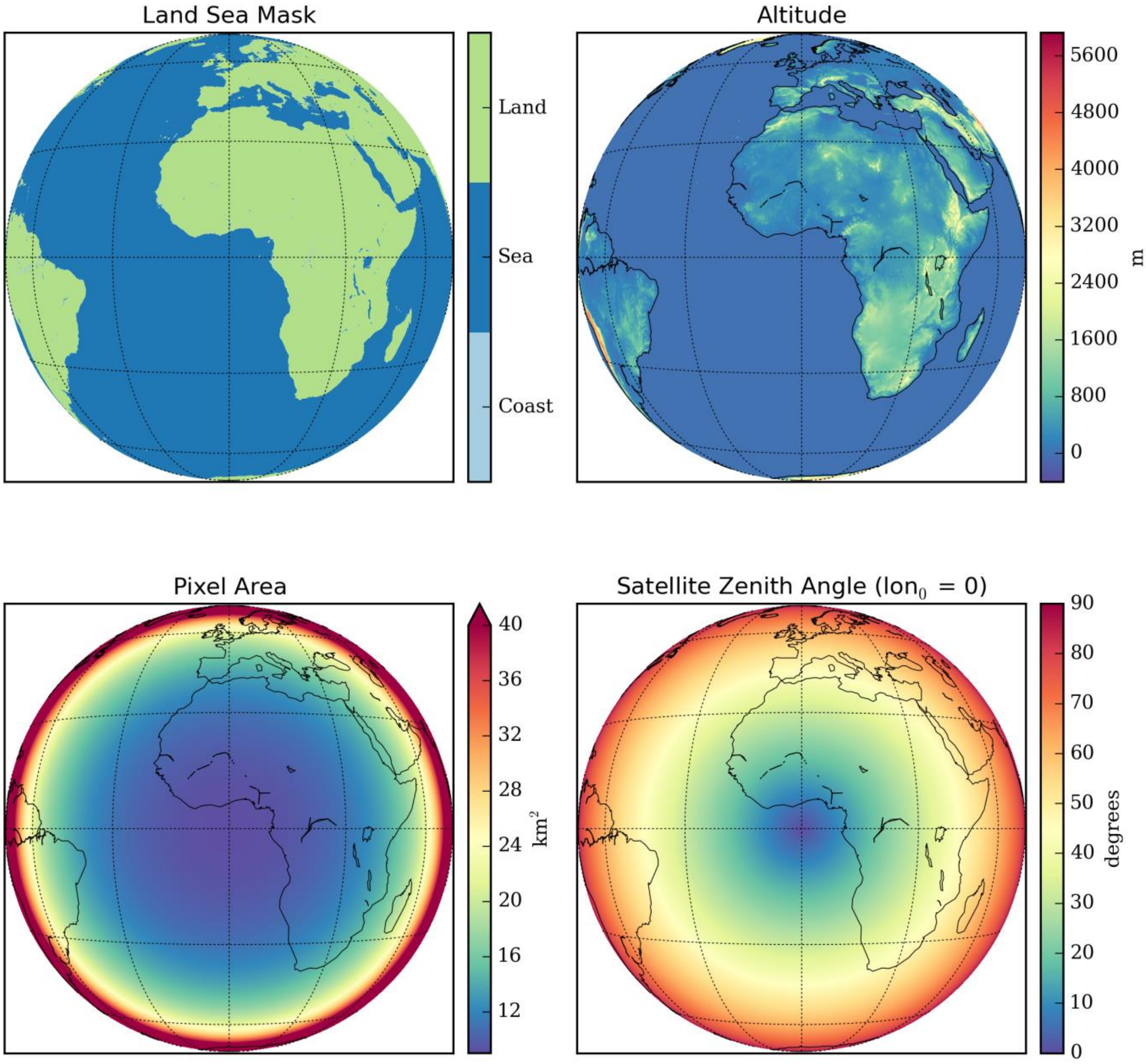

Figure 2.1 Example plots of available level 2 auxiliary data.

3 Level 3 Auxiliary Data

Level 3 products are distributed on regular 0.05° and 0.25° grids. The corresponding auxiliary data are described in the following two sections. Land-Sea-Mask, altitude and pixel area are valid throughout the dataset. However, the satellite position changes over time so that satellite zenith angles are not constant. See section 4 for details.

3.1 Regular 0.05° Grid

The 0.05° auxiliary data are stored in the netCDF4 file `class2_level3_aux_data_005deg.nc`, Figure 3.1 shows example plots of its contents. Land-Sea-Mask and altitude were obtained by remapping the corresponding level 2 fields to the 0.05° grid using a nearest neighbour approach. Pixel area as well as satellite zenith angle were derived directly from the latitude/longitude coordinates.

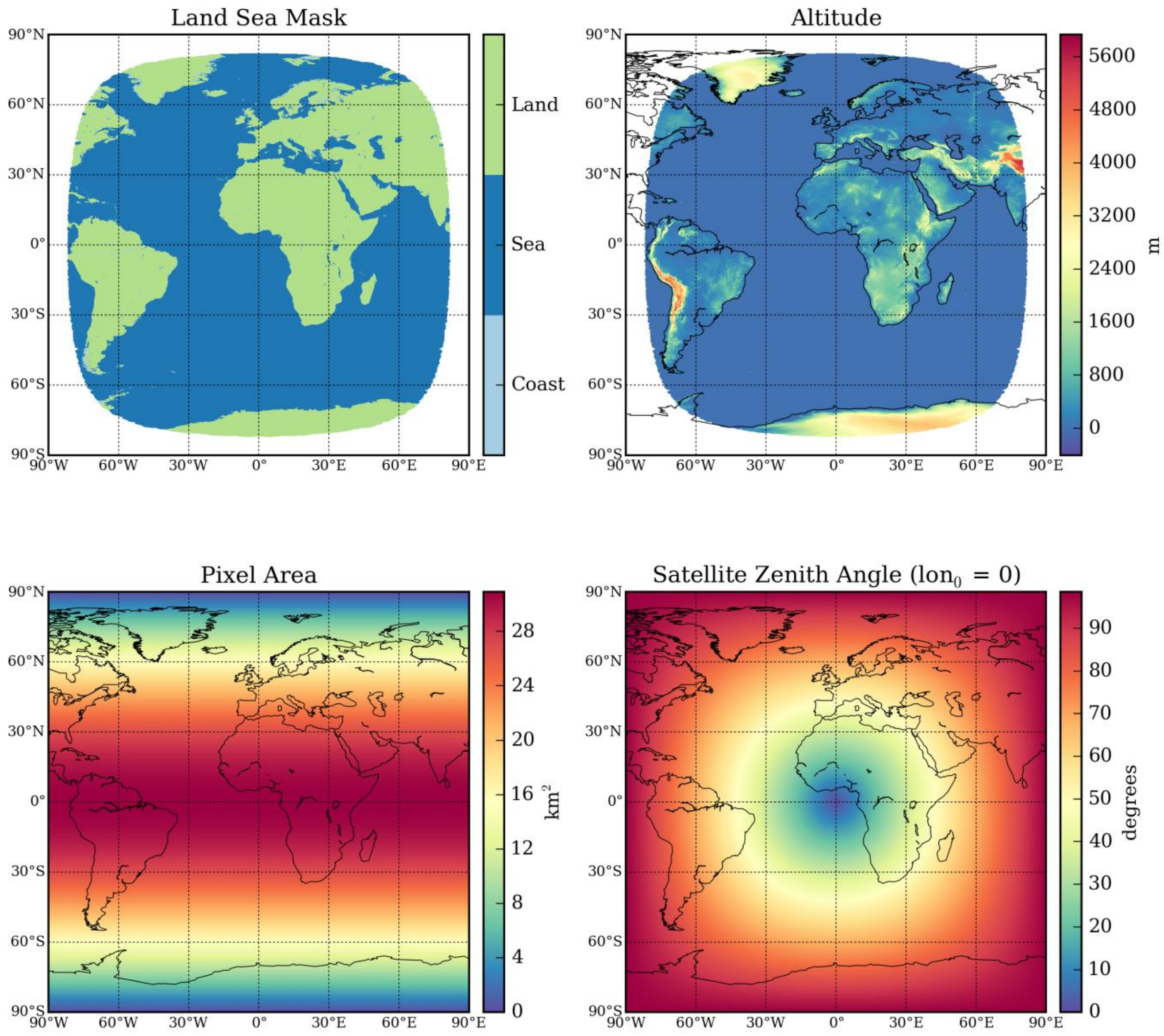

Figure 3.1 Example plots of available level 3 auxiliary data on the 0.05° grid.

3.2 Regular 0.25° Grid

The 0.25° auxiliary data are stored in the netCDF4 file `class2_level3_aux_data_025deg.nc`, its contents are shown in Figure 3.2. In contrast to the other grids, we provide mean quantities here (except for pixel area), which were obtained by area-weighted averaging of the 0.05° auxiliary data in 5x5 pixel boxes. This average is representing the 25 pixels more accurate than just the nearest neighbour. The discrete land sea mask is consequently transformed into a continuous land fraction. The pixel area was derived directly from the latitude/longitude coordinates.

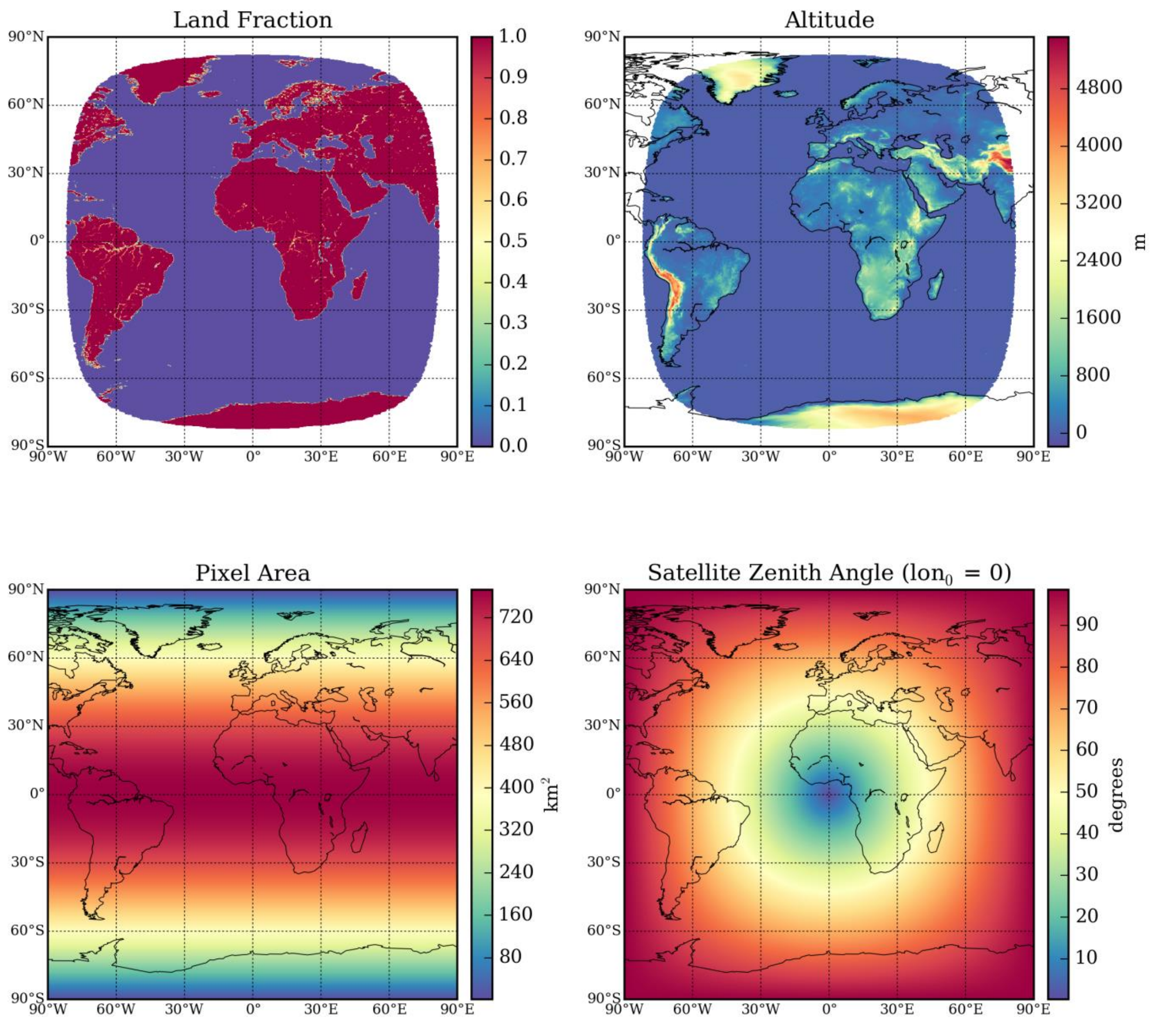

Figure 3.2 Example plots of available level 3 auxiliary data on the 0.25° grid.

4 Satellite Zenith Angles

During the CLAAS-2 time series several satellite maneuvers were performed (Figure 4.1). In contrast to the geographical coordinates, which can be reprojected, viewing angles depend on the satellite position. That is why we provide satellite zenith angles for the most prominent subsatellite points.

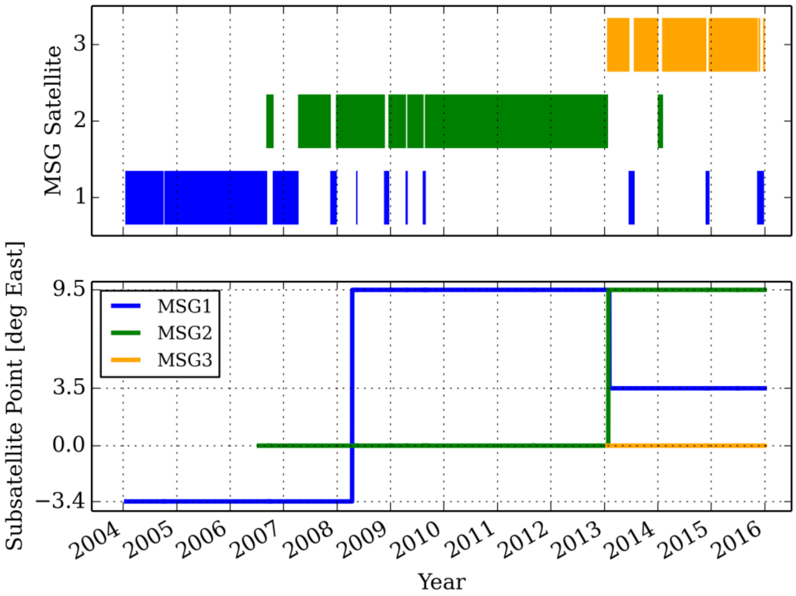

Figure 4.1: *Top*: Overview of data availability in CLAAS-2. Gaps are shown enlarged by a factor 5 for better visibility. *Bottom*: Most prominent MSG satellite positions used in CLAAS-2, based on the METEOSAT orbital parameters archive. Note that small scale variations around the mean positions were not taken into account.

In order to facilitate an automatic readout of the correct subsatellite point, we have included the following variables into all auxiliary files (*msgx* stands for *msg1*, *msg2* or *msg3*):

- `lon_0`: Set of most prominent subsatellite points: (-3.4, 0.0, 3.5, 9.5).
- `satzen`: Satellite zenith angle for each subsatellite point in `lon_0`.
- `msgx_lon0_id`: Subsatellite points occupied by MSGX. Contains indices referring to the coordinate variable `lon_0`. Example: `msgx_lon0_id = (0, 2)` means that MSGX occupied the subsatellite points `lon_0[0]` and `lon0[2]`, e.g. -3.4 and 3.5 degrees east. Note that we use zero-based indexing here.
- `msgx_lon0_time_bounds`: Specifies when MSGX occupied a certain position. Provides a tuple (start date, end date) for each subsatellite point in `msgx_lon0_id`.

Here is a step-by-step guide how to readout the subsatellite point and the corresponding satellite zenith angles at a certain point *t* in time:

1. Obtain the platform at a certain point in time by reading out the global *platform* attribute in a CLAAS-2 product. Level 3 products may have been generated using data from multiple platforms. If that is the case and you require the satellite zenith angle for your analysis, we recommend using level 2 data instead. Alternatively, at least for the daily mean products, you can find the contribution of each platform in the global *CMSAF_platform_and_timeslots* attribute.
2. Let the platform you obtained in the previous step be *msgx* (one of *msg1, msg2, msg3*). Find the time interval including the time of interest *t*, i.e. find the index *`idx`* so that `msgx_lon0_time_bounds[idx, 0] <= t <= msgx_lon0_time_bounds[idx, 1]` .
3. The subsatellite point of *msgx* at time *t* is `lon0[msgx_lon0_id[idx]]` . The corresponding satellite zenith angles are `satzen[msgx_lon0_id[idx], :, :]` .

Note that we assume row-major, zero-based indexing (C-like) here. Below you find an IDL and Python snippet for determining the subsatellite point as well as the corresponding satellite zenith angles of MSG1 at 2009-07-01 12:15. We use level 2 aux data in this example, but the method also works with level 3 aux data.

4.1 CLAAS ICDR

With the upcoming CLAAS ICDR (Interim Climate Data Record, the near-real-time continuation of CLAAS-2), both platform and subsatellite point will be stored in the level 2 products (variables `platform` and `lon0`) which makes finding the correct viewing angles much easier.

4.2 Python Example

```
import netCDF4
import datetime

# Read MSG1 aux data
ds = netCDF4.Dataset('claas2_level2_aux_data.nc')
msg1_lon0_id = ds.variables['msg1_lon0_id'][:]
msg1_lon0_time_bounds = ds.variables['msg1_lon0_time_bounds'][:]
lon0_vals = ds.variables['lon0'][:]

# Choose point in time
t = netCDF4.date2num(
    datetime.datetime(2009, 7, 1, 12, 15),
    calendar=ds.variables['msg1_lon0_time_bounds'].calendar,
    units=ds.variables['msg1_lon0_time_bounds'].units
)

# Determine subsatellite point
num_pos = len(msg1_lon0_id) # Number of positions occupied by MSG1
idx = 0 # Index of matching time interval
for ipos in range(num_pos):
    if msg1_lon0_time_bounds[ipos, 0]<=t<msg1_lon0_time_bounds[ipos, 1]:
        # Found the matching time interval!
        break
    else:
        idx += 1

if idx == num_pos:
    # No matching time interval found
    raise ValueError('Timestamp is either pre-launch or outside '
                     'the temporal coverage of the dataset')

lon0 = lon0_vals[msg1_lon0_id[idx]]
satzen = ds.variables['satzen'][msg1_lon0_id[idx], :, :]
print "Subsatellite point: {0} degrees east".format(lon0)
```

4.3 IDL Example

```
pro claas_lon0

; Read aux data
fileID = ncdf_open('claas_level2_aux_data.nc')

varID = NCDF_Varid(fileID,'msg1_lon0_id')
NCDF_Varget, fileID, varID, msg1_lon0_id

varID = NCDF_Varid(fileID,'msg1_lon0_time_bounds')
NCDF_Varget, fileID, varID, msg1_lon0_time_bounds

varID = NCDF_Varid(fileID,'lon0')
NCDF_Varget, fileID, varID, lon0_vals

varID = NCDF_Varid(fileID,'satzen')
NCDF_Varget, fileID, varID, satzens
NCDF_Close, fileID

; Choose point in time (time unit is 'days since 1970-01-01 00:00')
t = julday(7,1,2009,12,15) - julday(1,1,1970,0,0)

; Determine subsatellite point
num_pos = n_elements(msg1_lon0_id) ; Number of positions occupied by MSG1
idx = 0 ; Index of matching time interval
for ipos = 0,num_pos-1 do begin
    if ( t ge msg1_lon0_time_bounds[0,ipos] and $
         t lt msg1_lon0_time_bounds[1,ipos] ) then begin
        break
    endif else begin
        idx++
    endelse
endfor

if idx eq num_pos then begin
    print,'Timestamp is either pre-launch or outside the '
    print,'temporal coverage of the dataset'
endif else begin
    lon0 = lon0_vals[msg1_lon0_id[idx]]
    satzen = satzens[*, *, msg1_lon0_id[idx]]
    print, "Subsatellite point (degrees east):"
    print, lon0
endelse
end
```

5 References

Finkensieper, Stephan; Meirink, Jan-Fokke; van Zadelhoff, Gerd-Jan; Hanschmann, Timo; Benas, Nikolaos; Stengel, Martin; Fuchs, Petra; Hollmann, Rainer; Werscheck, Martin. (2016): CLAAS-2: CM SAF CLoud property dAtAset using SEVIRI - Edition 2. Satellite Application Facility on Climate Monitoring. DOI:10.5676/EUM_SAF_CM/CLAAS/V002. http://dx.doi.org/10.5676/EUM_SAF_CM/CLAAS/V002